Clouds and the Earth's Radiant Energy System (CERES)

Data Management System

Software Design Document

Cloud Retrieval
(Subsystems 4.1 - 4.3)

Architectural Draft

Chris Currey[1], Alice Fan[2], Timothy Murray[2],
Sunny Sun-Mack[2], and Carol Tolson[2]

[1]NASA Langley Research Center
Mail Stop 423
Hampton, VA 23681-0001

[2]Science Applications International Corporation (SAIC)
One Enterprise Parkway, Suite 300
Hampton, Virginia 23666

Atmospheric Sciences Division
NASA Langley Research Center
Hampton, Virginia 23681-0001

Release 1.0
March 1996

Preface

The Clouds and the Earth's Radiant Energy System (CERES) Data Management System supports the data processing needs of the CERES science research to increase understanding of the Earth's climate and radiant environment. The CERES Data Management Team works with the CERES Science Team to develop the software necessary to support the science algorithms. This software, being developed to operate at the Langley Distributed Active Archive Center (DAAC), produces an extensive set of science data products.

The Data Management System consists of 12 subsystems; each subsystem represents a stand-alone executable program. Each subsystem executes when all of its required input data sets are available and produces one or more archival science products.

The documentation for each subsystem describes the software design at various stages of the development process and includes items such as Software Requirements Documents, Data Products Catalogs, Software Design Documents, Software Test Plans, and User's Guides.

This version of the Software Design Document records the architectural design of each Subsystem for Reslease 1 code development and testing of the CERES science algorithms. This is a PRELIMINARY document, intended for internal distribution only. Its primary purpose is to record what was done to accomplish Release 1 development and to be used as a reference for Release 2 development.

TABLE OF CONTENTS

List of Figures

List of Tables

1.0 Introduction

The Clouds and the Earth's Radiant Energy System (CERES) is a key component of the Earth Observing System (EOS). The CERES instruments are improved models of the Earth Radiation Budget Experiment (ERBE) scanner instruments, which operated from 1984 through 1990 on the National Aeronautics and Space Administration's (NASA) Earth Radiation Budget Satellite (ERBS) and on the National Oceanic and Atmospheric Administration's (NOAA) operational weather satellites NOAA-9 and NOAA-10. The strategy of flying instruments on Sun-synchronous, polar orbiting satellites, such as NOAA-9 and NOAA-10, simultaneously with instruments on satellites that have precessing orbits in lower inclinations, such as ERBS, was successfully developed in ERBE to reduce time sampling errors. CERES will continue that strategy by flying instruments on the polar orbiting EOS platforms simultaneously with an instrument on the Tropical Rainfall Measuring Mission (TRMM) spacecraft, which has an orbital inclination of 35 degrees. In addition, to reduce the uncertainty in data interpretation and to improve the consistency between the cloud parameters and the radiation fields, CERES will include cloud imager data and other atmospheric parameters. The first CERES instrument is scheduled to be launched on the TRMM spacecraft in 1997. Additional CERES instruments will fly on the EOS-AM platforms, the first of which is scheduled for launch in 1998, and on the EOS-PM platforms, the first of which is scheduled for launch in 2000.

1.1 Document Overview

The CERES "Determine Cloud Properties, TOA and Surface Fluxes" Subsystem (Subsystem 4.0) has been divided into two separate subsystems. They are the "Determine Cloud Properties" Subsystem and the "Determine TOA and Surface Fluxes" Subsystem. The "Determine Cloud Properties" Subsystem has been further divided into two major components. Each component develops and publishes separate design documents, and develops separate sets of executable code. The first major component (Cloud Retrieval) combines the functions described in Algorithm Theoretical Basis Document (ATBDs) 4.1, 4.2, and 4.3, and provides the cloud masks and cloud properties for the cloud imager data on a per-pixel basis. The second major component is Convolution of Imager Cloud Properties with CERES Footprint Point Spread Function, corresponding to ATBD 4.4. The Determine TOA and Surface Fluxes Subsystem (ATBDs 4.5 and 4.6) completes Subsystem 4.0 and the SSF archival product by producing fluxes at the top-of-the-atmosphere (TOA) and fluxes at the surface.

This document addresses only the design for the Cloud Retrieval process. The intended audience for this document is the CERES Cloud Subsystem Teams, Subsystem testers, neighboring Subsystem teams, and science reviewers.

The objective of this document is to provide Release 1 architectural design specifications which were used to guide the development of the Cloud Retrieval component. The Release 1 design document addresses the requirements from the Science Team's ATBDs and the Software Requirements Document (SRD) 4.1 - 4.3 (Reference 1). The design document is developed by the Data Management Cloud Subsystem Team. This document currently contains Sections 1.0 and

2.0. Section 3.0 will not be completed for publication until Release 2. Appendix B contains a listing of the data structues for the output products. Appendix C contains a listing of several primary internal data structures. Neither of these Appendices will be completed until Release 2. The document sections are:

- 1.0: Introduction
  - 1.1: Document Overview
  - 1.2: Subsystem Overview
  - 1.3: Key Concepts
  - 1.4: Implementation Constraints
  - 1.5: Design Approach
- 2.0: Architectural Design
  - 2.1 Input Data Sets: InitCloud Routine
  - 2.2 Chunk: RetrieveChunk Routine
  - 2.3 Algorithm Managers
  - 2.4 OutputDataSets: OutputCloudData Routine
- References
- Appendix A - Abbreviations and Acronyms
- Appendix B - External Interfaces
- Appendix C - Data and Constants

The following sections are not included in this release:

- 3.0: Module Specifications (Detail Design)
- Appendix D - Error Messages
- Appendix E - Design Chart Symbols

1.2 Subsystem Overview

The Cloud Retrieval Subsystem's objective is to use high spectral and spatial resolution cloud imager data to determine cloud microphysical and optical properties. This provides a set of pixel cloud properties that are mapped onto the CERES footprint in the next process, documented in Software Design Document (SDD) 4.4. The major Cloud Retrieval science requirements are illustrated in Figure 1-1 and include:

1. Prepare a "chunk" of pixels (multiple scan lines of imager data): attach the imager radiances and various ancillary data to each imager pixel within the chunk. Classify each pixel as clear, cloudy, or uncertain. The pixel classification process uses various tests on the imager radiometric data and ancillary data to determine a cloud mask (Reference 2).

2. Determine cloud macrophysical properties (cloud layer and cloud top pressure) for cloudy pixels (ATBD 4.2) (Reference 3).

3. Determine cloud microphysical and optical properties (base and effective radiating center temperature and pressure, phase, particle size, optical depth at 0.65 micron, water/ice path, emittance at 10.8 micron) for cloudy pixels (ATBD 4.3) (Reference 4).

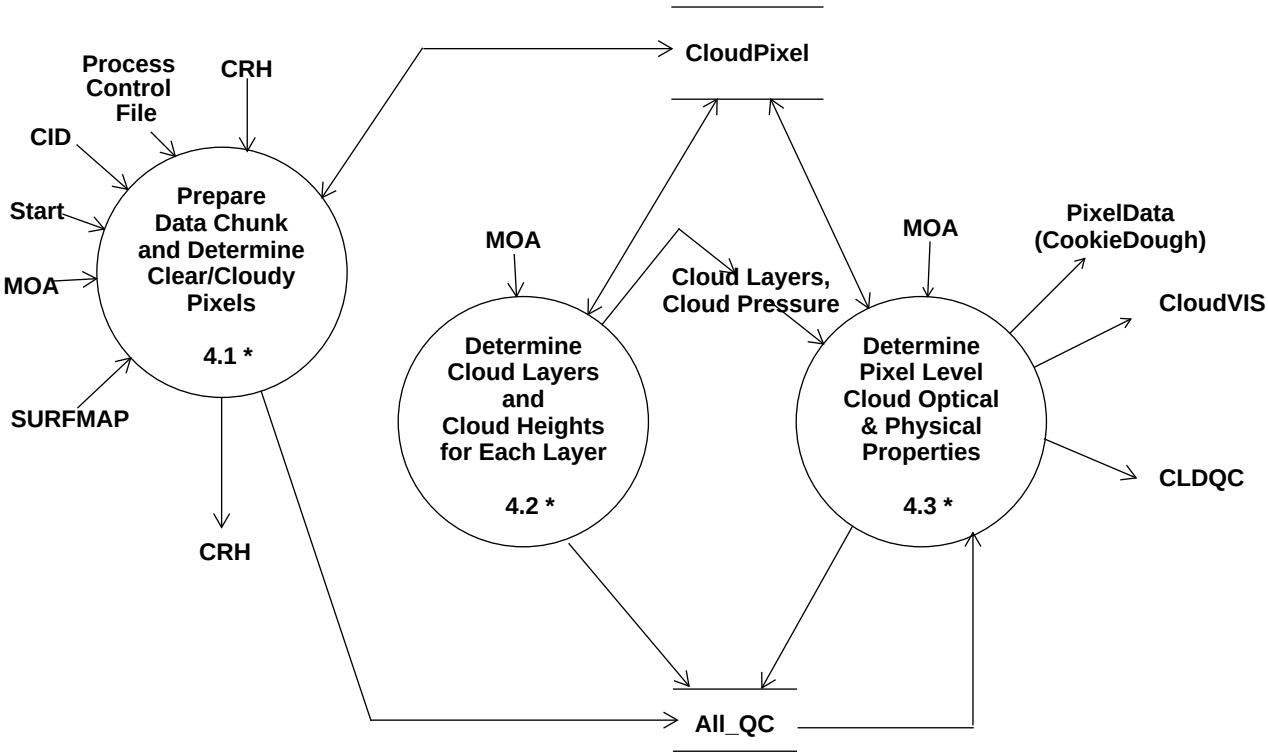

Figure 1-1. Subsystem 4.1 - 4.3 High-level Data Flow Diagram

The primary input data sets for the Cloud Retrieval Subsystem are:

1. The Cloud Imager Data (CID) data product contains time code, pixel location, viewing geometry, and radiance data. For the TRMM mission, CERES will use the Visible Infrared Scanner (VIRS) cloud imager data. For the next launches on EOS AM and PM spacecraft, CERES will use selected channels from the Moderate Resolution Imaging Spectrometer (MODIS) imager data. The Release 1 test data are Advanced Very High Resolution Radiometer (AVHRR) imager data from the NOAA-9 spacecraft.

2. The SURFace MAP (SURFMAP) data product is a set of maps for elevation, water content, ERBE Scene ID, snow, ice, ecosystem, and a condensed ecosystem/terrain map on a 10-

minute equal-angle grid. The condensed map is built by an off-line process using information from a coastline map, a surface terrain map, and the ecosystem map.

3. The Meteorological, Ozone, and Aerosol (MOA) (formerly named ASTR) data product contains meteorological data (surface temperature and surface pressure; 58 atmospheric levels of temperature, humidity, wind, and height; precipitable water, 26 levels of ozone, column ozone, and aerosols on a 1.25-degree equal-area grid.)

4. The Release 1 Clear Radiance History (CRH) data product contains albedo, brightness temperature, and the cosine of the solar zenith angle on a 10-minute equal-angle grid.

The data sets which are input to science algorithms were acquired and/or developed by the Cloud Working Group. A Process Control File (PCF) specifying file names and run-time parameters is prepared by the Distributed Active Archive Center (DAAC) personnel. PCFs are prepared by the development team during development for testing purposes and are part of the DAAC delivery package.

The output products are the pixel-based cloud properties needed by the convolution process (Table 4.4 in ATBD 4.4), a validation product, Status Message Facility (SMF) log files, and Quality Control (QC) reports. The hourly output products are described in greater detail in the Data Products Catalog (Reference 5).

1.3 Key Concepts

This section discusses the major concepts behind the design of the Cloud Retrieval Subsystem.

- *Imager pixel*
- *Data chunk*
- *Bounding rectangle*
- *Master pixel*
- *Chunk loop*
- *Algorithm loop*
- *The framework*
- *Quality edit checks*

Imager pixel refers to a single cloud imager field-of-view, which ranges from 0.25 - 1 km for MODIS pixels, 2 km for VIRS pixels, and 4 km for AVHRR-GAC (Global Area Coverage) pixels. Some of the algorithms process one pixel at a time and some of the algorithms process arrays of pixels at a time.

Data chunk: The Cloud Retrieval Subsystem processes a chunk of imager data at a time. A chunk is a selected number of imager scanlines, which must be at least as large as the largest array of data any science algorithm needs.

Bounding rectangle: After reading the imager data chunk, the diagonal corner latitudes and longitudes of a rectangle that circumscribes the imager chunk are computed. These "bounding" latitude and longitude pairs are used to input a corresponding set of grids from the various input surface characteristic maps, the gridded MOA data set, and the gridded CRH data set. These data form part of a set of attributes that are associated with each imager pixel; each pixel is known as ’master’.

Figure 1-2 shows an arbitrary satellite orbit swath with an illustration of a data chunk and the bounding rectangle.

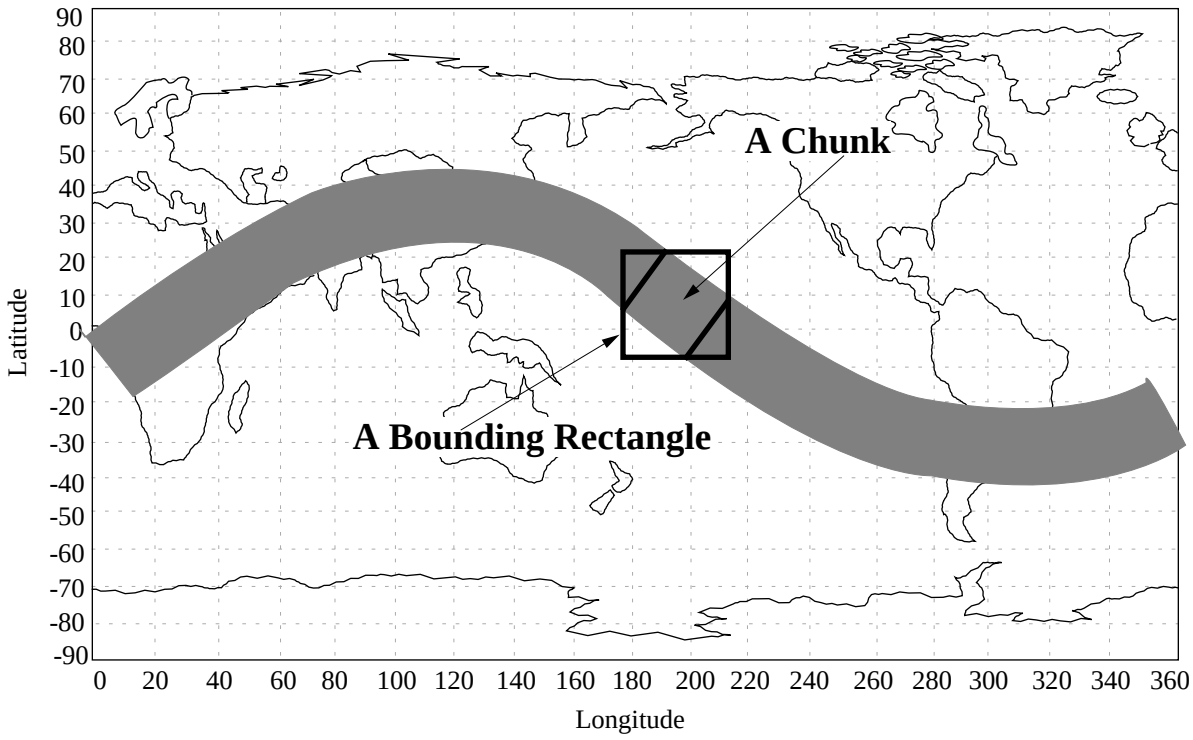

Figure 1-2. Chunk and Bounding Rectangle

Master pixel: The framework prepares the master pixel by attaching parameters from input data, from calculated parameters, and from science algorithm results. The master pixel serves as a resource from which to select only the attributes a particular science algorithm requires. The master pixel consists of imager radiances and reflectances, the surface characteristics, the geotype, solar angles, clear-sky historical data, various flags, pointers to a set of 3-dimensional meteorological profiles, and algorithm results.

Chunk loop: The Cloud Retrieval Subsystem processes one chunk of imager data at a time. The chunk loop continues until there are insufficient data in the imager input hourly file to form the last chunk. The leftover scanlines will be processed with additional scanlines from the next hourly input file in Release 2.

Algorithm loop: Surrounding each science algorithm is an input interface routine that does the tailoring from the master pixel for the algorithm. On the output side of each algorithm is another interface routine that receives the algorithm results and attaches it to the master pixel. If a particular algorithm has been selected to execute, an interface is prepared, an algorithm is executed, and results are stored. The code loops through all of the science algorithms that are selected for a particular run.

The framework interfaces with the input data, the algorithms, and the output data. The framework accesses and prepares input data for each science algorithm, executes the algorithm, and collects the results. The framework initializes the output files and writes results as processing proceeds. The framework provides the flexibility to add, replace, or delete a particular contributed science algorithm and to selectively execute it.

Quality edit checks for bad input data are made and fill data are used if the input data within an imager scanline are unusable. No fill scanlines are provided when entire scanlines are missing. Limit checks to ensure that the input data are within reasonable limits are implemented. Data that are outside these limits are excluded from further processing, and a diagnostic message is generated. Quality checks on science results and within the algorithm, along with science algorithm error-handling, will be evolving throughout Release 2.

1.4 Implementation Constraints

The implementation constraints are the EOSDIS operating environment, the Science Data Production (SDP) Toolkit (Reference 6), the limits of computer CPU, data throughput, network capacity, and system complexity. In addition, it is an EOSDIS requirement that all filenames must be obtained from the Process Control File by using a Toolkit call, and all file opens must be accomplished by using a Toolkit call. Currently some of the input data files are managed by the science algorithms. Correct file names and open statement usage will be handled on a case-by-case basis with the algorithm developer.

The cloud framework is written in FORTRAN 90 and interfaces with science algorithms written in either FORTRAN or C. The science algorithms are contributed code from members of the Cloud Working Group. Therefore, it was necessary to implement the framework around the contributed code and to consider the contributed code as "black boxes." The approach was taken to interface with the science algorithms by preparing tailor-made interfaces to the black boxes.

The Cloud Subsystem needs hourly input files and produces hourly output products for a single Product Generation Executable (PGE). It is assumed that the Planning and Data Production System (PDPS) will not start the cloud processing system until all required input files are available and that the PDPS will stage all input data files needed for the run.

Release 1 was delivered to the LaRC Distributed Active Archival Center early March 1996 and was run within the EOSDIS Interim Release 1 system. The code is designed to process global data from the existing ERBE/AVHRR data from the NOAA-9 and 10 spacecraft.

The validation products will be written in Hierarchical Data Format (HDF) early in the Release 2 development. For Release 1, only common parameters for the hourly products are collected and output in the form of a header record. These metadata include general header information such as instrument, satellite, start data, start time, end data, end time, and processing date and time. EOSDIS requires that metadata be written using metadata toolkit calls. The metadata toolkit will be addressed in Release 2.

1.5 Design Approach

The Cloud Retrieval Subsystem was developed using an object-based design approach. Isolating science algorithms from the framework and subsetting the functionality into objects was suited to the object oriented approach. FORTRAN 90 module constructs made the object based approach easy to implement.

The Cloud Retrieval Subsystem is a collaborative effort between the CERES Science Team and the CERES Data Team members. Development began with a prototype system following the requirements from the SRD and from Dr. Bryan Baum. The idea was to quickly produce a framework in the C programming language to provide a test bed for the science algorithms, which were delivered six to nine months later to the Science Team. The Science Team integrated algorithms, tailored the interfaces, and checked out the algorithms.

The data team developed the system in three iterative builds, where each build was designed, coded and tested. The first build concentrated on inputting all the input data, calculating additional parameters, building the master pixel for each pixel in the chunk, and outputting the science and validation products. Completion of the first build culminated in the Subsystem end-to-end milestone test.

The second build concentrated on interacting with the Science Team members to flesh out the details for their algorithm interfaces, and to tailor the science algorithm from a stand-alone mode to one which would receive its data from the framework. During the second build, the science algorithms were integrated into the framework, tested using the master pixel data as the input data source for the algorithm, and results output to the master pixel and then to the output products. After completing the first two production builds, extensive difference testing between production results and prototype results uncovered errors in both versions. At the end of Build 2 of the production code, all team members concentrated on the production version, and the prototype was archived.

The third build concentrated on preliminary quality control reports and statistics, contributed routine upgrades, and finalizing the CloudVIS product. CloudVIS is used specifically as input to a visualization tool, Satellite Image Visualization (SIVIS). CloudVis is also input into a subsetting postprocessor and an IDL visualization program developed for validation.

Prototype development allowed for early science algorithm integration and provided insight into solving design problems inherent in providing a framework for many different science algorithms in various languages. About 50% of the contributed code is in some version of FORTRAN while the other 50% is in C. All FORTRAN code was compiled using the FORTRAN 90 NAG compiler.

Switching to FORTRAN 90 had the advantage of a fresh look at the design after considerable experience with the prototype. The FORTRAN 90 compiler has more rigid constraints and, therefore, found errors that the C and the FORTRAN 77 compilers did not. The difference testing revealed errors in the prototype and production versions of the code.

2.0 Architectural Design

This section discusses two high-level views of the Cloud Retrieval Subsystem. The dynamic view is shown in the flowchart in Figure 2-1. The static view is shown in the context diagram in Figure 2-2. The design approach of providing a framework that serves data to the algorithms and manages algorithm results is shown in Figure 2-1. Processing proceeds through two major loops. The outer chunk loop processes pixels in scanlines in a chunk until all chunks in the hour imager data are processed. For each chunk, the three steps in the algorithm loop are processed for each algorithm that has been selected for the particular job.

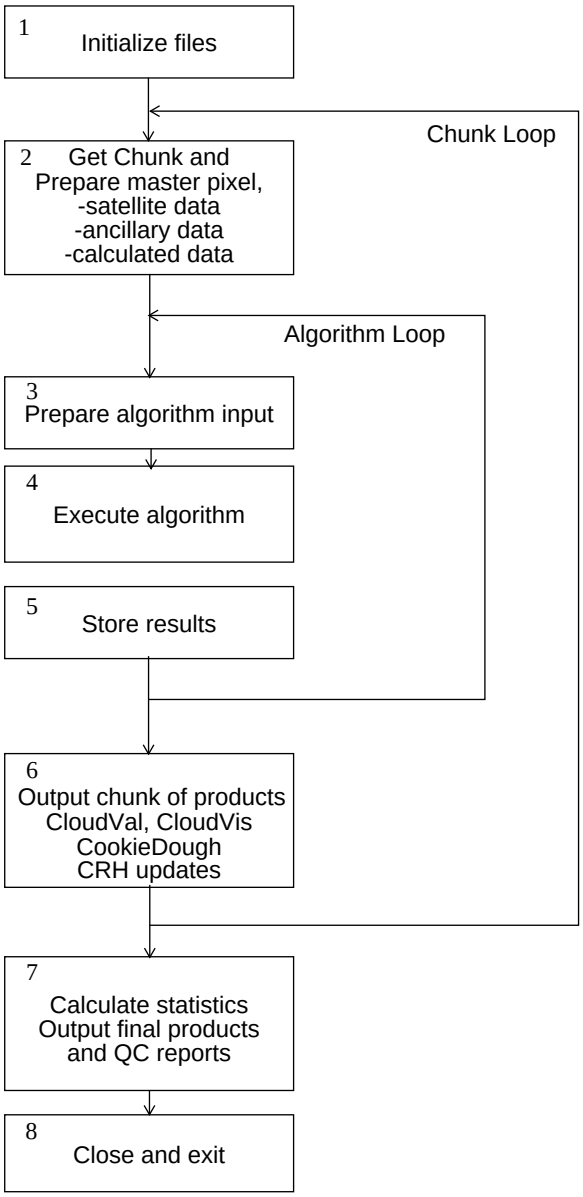

Figure 2-1. High-level Cloud Retrieval Flow Chart

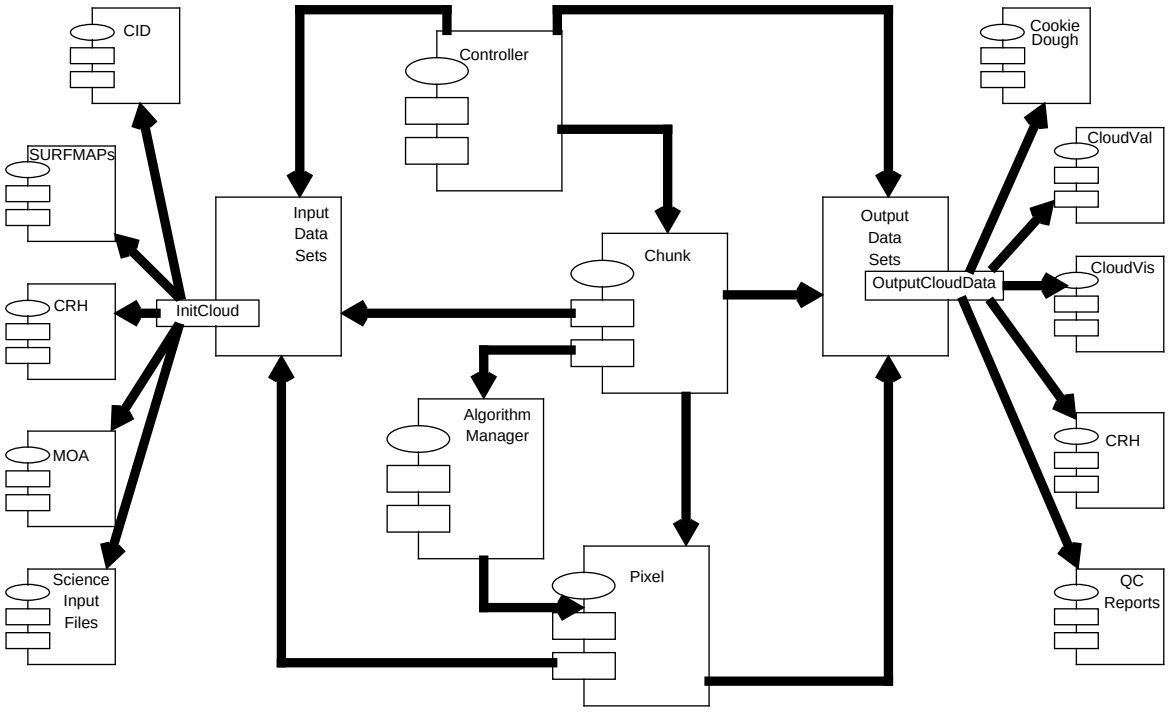

Figure 2-2. Design Overview Context Diagram

The major steps and logic are

1. Initialize the cloud process, get the filenames, open the input files, check all input file header records, and open the output files.

   Steps 2 through 6 are the outer chunk loop. Process all chunks within the hour.

2. Prepare a "chunk" (multiple scan lines of CID imager data) of master pixels: Attach the imager radiances and reflectances, location data, various ancillary input surface conditions, clear-sky history information, and pointers to the meteorological profiles to each pixel within the chunk. The parameters in the pixels, CloudPixel, are listed in Appendix C .

   Steps 3 through 5 are the inner algorithm loop: For each algorithm selected:

3. Prepare tailored data structures for the algorithm by extracting the required parameters from the master pixel.

4. Invoke a science algorithm chosen from the following major catagories

a) Classify each pixel as clear, cloudy, or uncertain. The pixel classification process uses various tests on the imager radiometric data and ancillary data to determine a cloud mask (ATBD 4.1) (Reference 3). The Release 1 algorithms are Baum cloud mask algorithm and the Welch snow/ice, desert, smoke, and cloud mask algorithms.

b) Determine cloud macrophysical properties (cloud layer and cloud top pressure) for cloudy pixels (ATBD 4.2) (Reference 4). The Release 1 algorithms are a partial Coakley Spatial Coherence algorithm and the Baum Classification algorithm.

c) Determine cloud microphysical and optical properties (base and effective radiating center temperature and pressure, phase, particle size, optical depth at 0.65 micron, water/ice path, emittance at 10.8 micron) for cloudy pixels (ATBD 4.3) (Reference 5). The Release 1 algorithms are the Stowe aerosol algorithm, the Minnis VINT algorithm, and the Platnick algorithm.

5. Collect the algorithm output and store on the master pixel structure.

6. Output the pixel cloud properties, update the clear-sky map, and start the next chunk. The parameters in Pixel_Data (refered to as CookieDough), CloudVal, and CloudVis are listed in Appendix B.

7. At the end of the processing, prepare a processing summary report, shutdown the algorithms, and

8. Close the files, and terminate processing.

Figure 2-2, the context diagram, shows the major components of the Cloud Retrieval Subsystem. All of the input data sets are dealt with in the InputDataSets component. This corresponds to Step 1 discussed below. The components that deal with the main chunk loop are Chunk, Pixel, and Algorithm Manager. Chunk and Pixel together correspond to Step 2 below, and in addition, provide data to the algorithms. The Algorithm Manager is an abstraction of individual managers that prepare algorithm-specific input, call the algorithm, and manage results according to the type of algorithm invoked. These are steps 3, 4, and 5. Similar to the input, OutputDataSets represents the functions to get pixel data from the chunk and output them to the various products (Step 6) and the functions to calculate final statistics for the Quality Control (QC) reports (Step 7). Step 8 closes all files and shuts down the processing. Each of the major components are shown in more detail and discussed in the following sections.

2.1 Input Data Sets: InitCloud Routine

The InitCloud routine drives the initialization process. EOSDIS Science Data Production (SDP) Toolkit calls are made to read the input file names from the Processing Control File (PCF). If the files exist, Toolkit calls are made to open the files. The smaller algorithm support input files are both opened and the data are read into memory at this time. If any of the input files could not be opened or do not exist, an error message is written to the Toolkit Status Message File (SMF), and processing terminates. If all input files were successfully opened, the output files selected in the PCF are opened for writing. The output file names are also read in from the PCF using Toolkit routines.

2.2 Chunk: RetrieveChunk Routine

The RetrieveChunk functions centered around processing a chunk's worth of data and building the master pixel are illustrated in Figure 2-3.

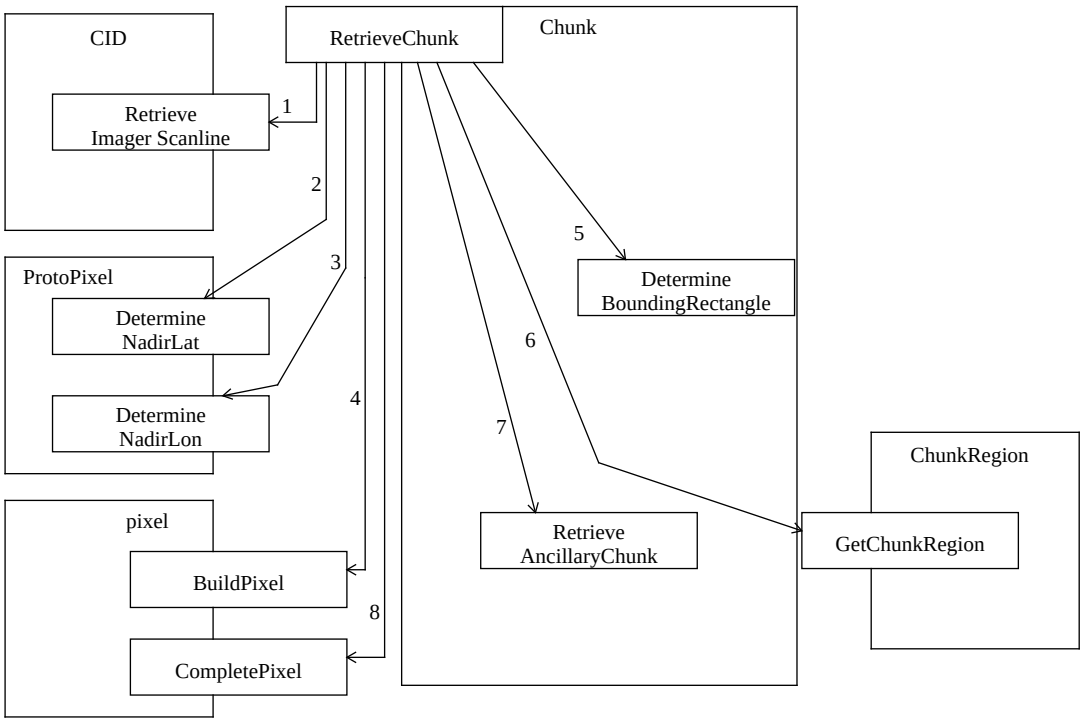

Figure 2-3. RetrieveChunk Scenario Diagram

RetrieveChunk is the framework work horse. Step 1 reads in a scanline of imager data and processing loops through the scanline to process each pixel in the scanline. All relevant imager data parameters are stored in the data structure, "protopixel" at the end of Step 1. The parameters in ProtoPixel are listed in Appendix C.

NadirLat and NadirLon are calculated in Steps 2 and 3.

Step 4, BuildPixel, calculates solar angles and gets the parameter values from the protopixel and attaches them to the master pixel. Additional detail for BuildPixel is shown in Figure 2-4. The first four steps process all pixels in the scanline, and then all scanlines in the chunk.

The bounding rectangle is determined in Step 5, and a chunk's worth of ancillary data within the bounding rectangle latitude and longitude boundaries is determined.

Steps 6 and 7 read in the chunk data (See Figure 2-5) based on the bounding rectangle specifications.

Step 8, CompletePixel, attaches parameters from the chunk data onto the master pixel (See Figure 2-6). Steps 5 through 8 complete the chunk processing.

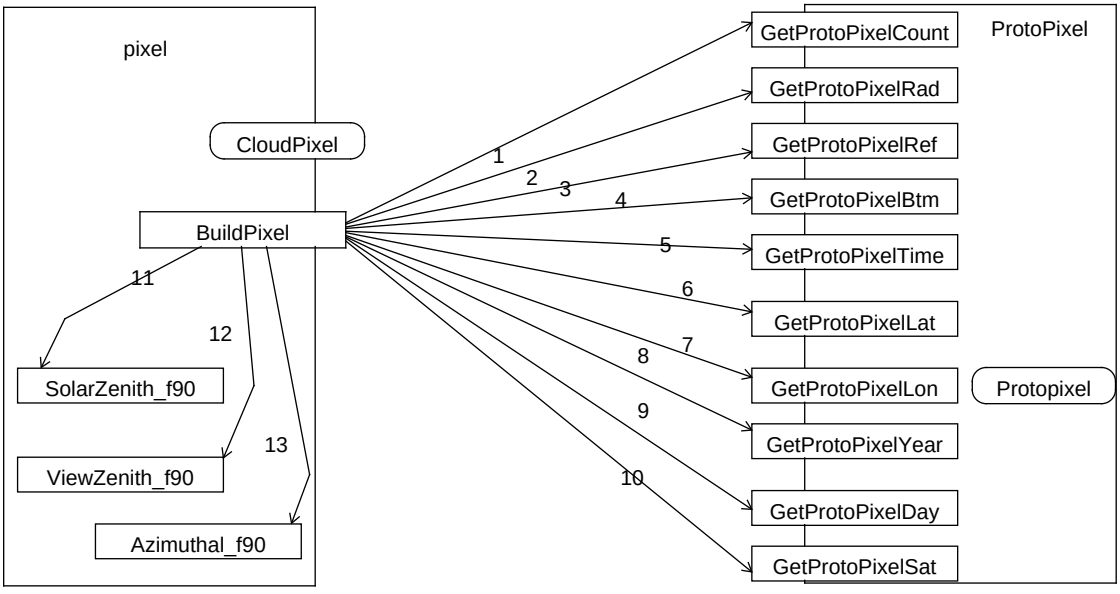

Figure 2-4. BuildPixel Scenario Diagram

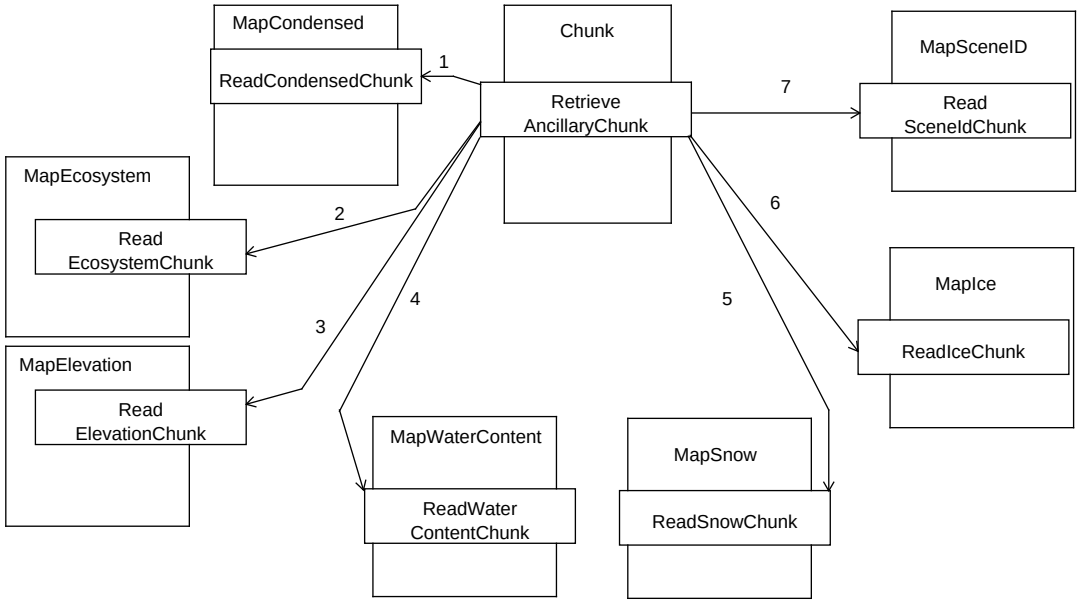

Figure 2-5. RetrieveAncillaryChunk Scenario Diagram

In the last step, CompletePixel, the ancillary data and derived quantities are attached to the master pixel. The master pixel now contains the parameters any of the science algorithms need along with fields reserved for the science results.

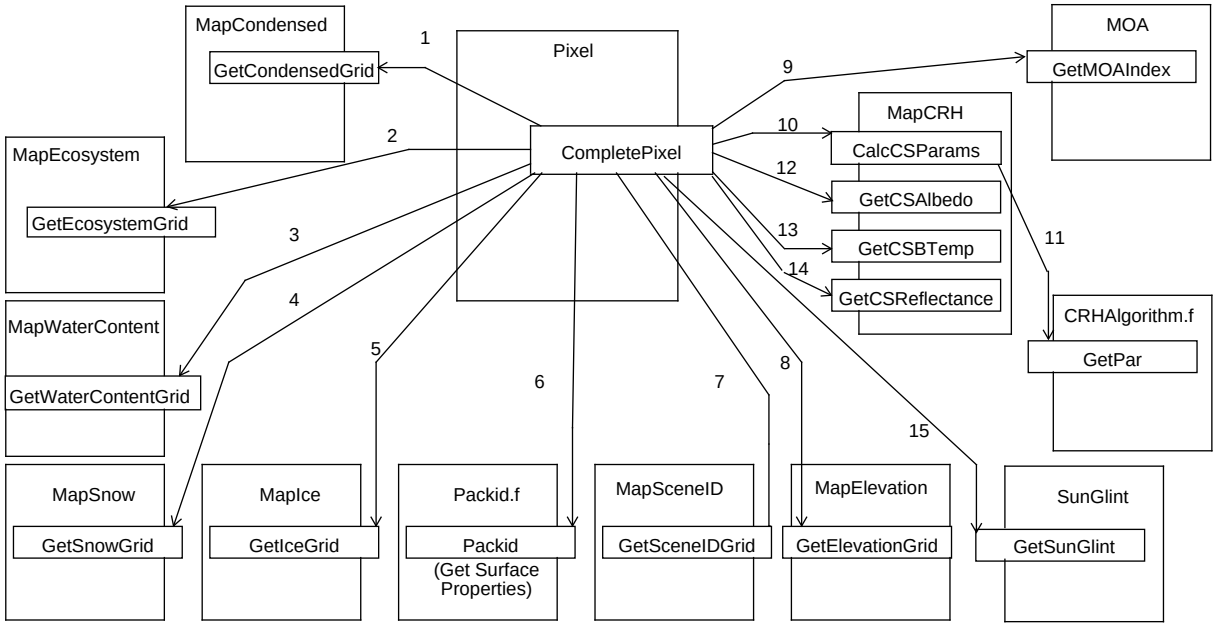

Figure 2-6. CompletePixel Scenario Diagram

2.3 Algorithm Managers

There are algorithm managers for the mask algorithms, the layer algorithms, and the optical properties algorithms. The algorithms are listed in Table 2-1. The managers read the ancillary science data that are specific to the particular algorithm. The manager modules are fairly similar in structure, and only one manager and one algorithm are discussed and illustrated in Figure 2-7.

Much of the code for the Cloud Retrieval Subsystem was contributed by members of the Cloud Working Group. The algorithms are discussed in the ATBDs. The Cloud Data Management Team designed and coded the framework and the interface routines to the Toolkit. Table 2-1 lists the algorithm name, the contributor, the developer, the source language, and the status of the code for Release 1. The same information is provided for the framework and Toolkit interface routines. Since the science algorithms are treated as "black boxes," no attempt has been made to discuss the design of these algorithms.

Table 2-1. Contributed Science Code (1 of 2)

Algorithm Name	Contributor/ Developer	Language	Release 1 Status
Cloud mask (CeresMask)	Dr. Bryan Baum Ms. Qing Trepte	C	Integrated
Polar, desert, smoke, cloud mask	Dr. Ron Welch Mr. Scott Berendes	C	Integrated
Spatial Coherence layer	Dr. Jim Coakley	Fortran 77	Partial algorithm delivered/ integrated
Cloud Classification	Dr. Bryan Baum Mr. Vasant Tovinkere Mr. Jay Titlow	C	Integrated
CO2 Slicing layer	Dr. Bryan Baum Ms. Qing Trepte	C	Algorithm development underway
VINT	Dr. Pat Minnis Mr. Pat Heck	Fortran 77	Integrated
Aerosol Optical Depth Retrieval	Dr. Larry Stowe	Fortran 77	Integrated
Platnick (Optical Thickness)	Dr. Steve Platnick	Fortran 77	Integrated
Air Mass, Meteorological analysis	Dr. Bryan Baum Mr. Jay Titlow	Fortran 77	Integrated
Surface property retrieval	Dr. Dave Kratz Dr. Tom Charlock Mr. Dave Rutan	Fortran 77	Integrated
CID-AVHRR	Dr. Kuo	C	Integrated

Table 2-1. Contributed Science Code (2 of 2)

Algorithm Name	Contributor/ Developer	Language	Release 1 Status
Input CRH	Mr. Pat Heck	Fortran 77	Integrated
Update CRH	Mr. Pat Heck	Fortran 77	Algorithm development underway
CERESlib	Ms. Maria Mitchum Mr. Joe Stassi	Fortran 90	Interfaces completed
Framework	Mr. Tim Murray Ms. Alice Fan Ms. Sunny Sun-Mack	Fortran 90	Complete
Toolkit Interface Routines	Ms. Alice Fan Ms. Sunny Sun-Mack	Fortran 90	Complete
Scripts	Mr.Tim Murray	Unix	Complete
ProcessControlFiles/Status Message Files	Ms. Alice Fan Ms. Sunny Sun-Mack	Unix, Tool-kit	Complete

The Manager checks a flag for each of the possible contributed algorithms. If the flag indicates that the algorithm has been selected to run, it calls the corresponding interface routine, as shown in Step 1 in Figure 2-7. In Step 2, the interface routine gets a master pixel, CloudPixel, and extracts only the parameters the particular algorithm needs. Step 3 calls the science algorithm. After the science algorithm completes its processing, the interface routine collects the science results, and Step 4 puts them on the master pixel. Figure 2-7 shows the MaskManager and the Baum CloudMask algorithm (CeresMask Interface) as an example of how the managers, the master pixel, and the algorithm work together.

.

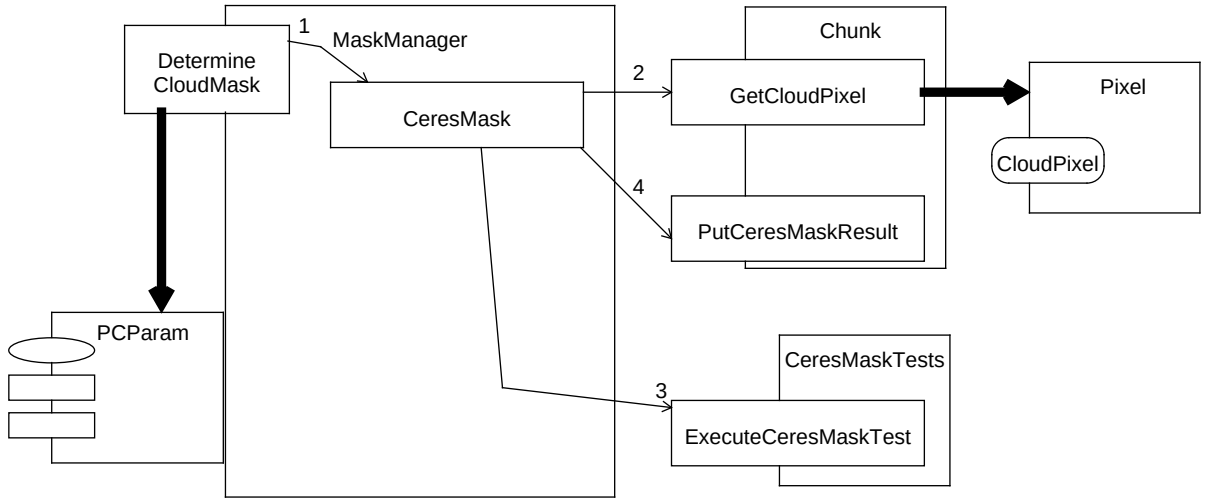

Figure 2-7. MaskManager Scenario Diagram

2.4 OutputDataSets: OutputCloudData Routine

The scenario diagram which illustrates writing the cloud data and algorithm results to the various products is shown in Figure 2-8.

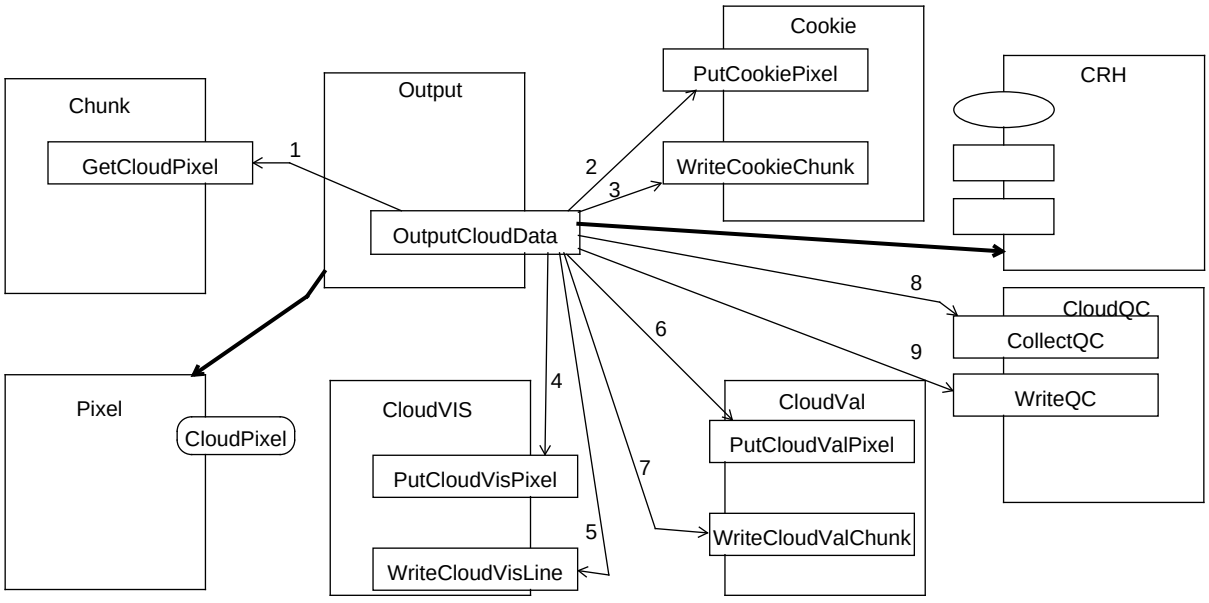

Figure 2-8. OutputCloudData Scenario

Each of the products shown in Figure 2-8 are only produced if they have been selected prior to run-time. These options are set in the Process Control File.

Figure 2-8 shows that Step 1 gets the master pixel in the routine, GetCloudPixel. Then, depending on which products have been requested, the particular data structures are formed by extracting the product-specific output parameters from the master pixel. The OutputCloudData routine tailors the output for each of the products, Cookie, CloudVis, and CloudVal. The updated CRH product algorithm will be integrated in Release 2.

The "Put.." routines put the pixel data into arrays and the "Write.." routines write the data to the product file. The data structures for these three output products are shown in Appendix B.

References

1. Clouds and the Earth's Radiant Energy System (CERES), Data Management System Software Requirements Document, Determine cloud Properties (Subsystems 4.1-4.4), Release 1, Version 1, April 1995

2. Clouds and the Earth's Radiant Energy System (CERES), Algorithm Theoretical Basis Document, Imager Clear-Sky Determination and Cloud Detection (System Volume 4.1), Release 1.1, April 1994

3. Clouds and the Earth's Radiant Energy System (CERES), Algorithm Theoretical Basis Document, Summary of the Cloud Pressure Retrieval Methods (System Volume 4.2), Release 1.1, April 1994

4. Clouds and the Earth's Radiant Energy System (CERES), Algorithm Theoretical Basis Document, Cloud Optical Property Retrieval (System Volume 4.3), Release 1.1, April 1994

5. CERES Data Management System Data Products Catalog. Release 1.0, August 1994.

6. Science Data Production Toolkit User's Guide for the ECS Project, August 1995.

Appendix A

Abbreviations and Acronyms

Appendix A - Abbreviations and Acronyms

ATBD	Algorithm Theoretical Basis Document
AVHRR	Advanced Very High Resolution Radiometer
CERES	Clouds and the Earth's Radiant Energy System
CID	Cloud Imager Data
CRH	Clear Radiance History
DAAC	Distributed Active Archive Center
EOS	Earth Observing System
EOS-AM	EOS Morning Crossing Mission
EOSDIS	EOS Data and Information System
EOS-PM	EOS Afternoon Crossing Mission
ERBE	Earth Radiation Budget Experiment
ERBS	Earth Radiation Budget Satellite
GAC	Global Area Coverage (AVHRR data mode)
HDF	Hierarchical Data Format
LaRC	Langley Research Center
MODIS	Moderate Resolution Imaging Spectrometer
MOA	Meteorological, Ozone, and Aerosol
NASA	National Aeronautics and Space Administration
NOAA	National Oceanic and Atmospheric Administration
PCF	Processing Control File
PDPS	Planning and Data Production System
PGE	Product Generation Executable
QC	Quality Control
SDD	Software Design Document
SDP	Science Data Production
SIVIS	Satellite Image Visualization
SMF	Status Message File
SRD	Software Requirements Document
SSF	Single Satellite CERES Footprint TOA and Surface Fluxes, Clouds
SURFMAP	Surface Properties and Map
TOA	Top-of-the-Atmosphere
TRMM	Tropical Rainfall Measuring Mission
VIRS	Visible Infrared Scanner

Appendix B

External Interface

Appendix B - External Interface

The data structure, CookiePixel, shows the data elements written to the file CookieDough.

CookiePixel (CookieDough - Table 4.4)

```
 integer, parameter :: NumLayers = 2
 integer, parameter :: TotNumChan = 11

 TYPE CookiePixel
    REAL                                :: TimeObs
    REAL                                :: colatitude
    REAL                                :: longitude
    REAL                                :: viewZen
    REAL                                :: azimuth
    REAL                                :: rad00_6
    REAL                                :: rad03_7
    REAL                                :: rad11_0
    REAL,    DIMENSION ( TotNumChan )   :: bidir

    REAL                                :: PreciWater
    REAL                                :: AerOptDepth_Strat
    REAL                                :: AerOptDepth_Total
    REAL                                :: EffRadius_Strat
    REAL                                :: EffRadius_Total
    REAL                                :: SkinTemp

    REAL,    DIMENSION ( NumLayers )    :: OptDepth_eff
    REAL,    DIMENSION ( NumLayers )    :: emissivity
    REAL,    DIMENSION ( NumLayers )    :: WaterPath
    REAL,    DIMENSION ( NumLayers )    :: TopPressure
    REAL,    DIMENSION ( NumLayers )    :: EffPressure
    REAL,    DIMENSION ( NumLayers )    :: EffTemp
    REAL,    DIMENSION ( NumLayers )    :: EffHeight
    REAL,    DIMENSION ( NumLayers )    :: BotPressure
    REAL,    DIMENSION ( NumLayers )    :: ParticleRadius
    REAL,    DIMENSION ( NumLayers )    :: AspectRatio
    INTEGER, DIMENSION ( NumLayers )    :: ParticlePhase

    INTEGER (int2)                      :: Algorithms
    INTEGER (int2)                      :: elevat
    INTEGER (int2)                      :: solzen
    INTEGER (int1)                      :: DayNightFlag
    INTEGER (int1)                      :: LandType
    INTEGER (int1)                      :: SeaType
    INTEGER (int1)                      :: Airmass
    INTEGER (int1)                      :: Sunglint
    INTEGER (int1)                      :: SnowIce
    INTEGER (int1)                      :: Layers
    INTEGER (int1)                      :: QAFlag
    INTEGER (int1)                      :: RutanEco
    INTEGER (int1), DIMENSION ( NumLayers )   :: CloudFraction
    INTEGER (int1)                      :: Spare_01
    INTEGER (int1)                      :: Spare_02
    INTEGER (int1)                      :: Spare_03
 END TYPE CookiePixel
```

The data structure, VisPixel, lists the data elements written to the file CloudVIS. The structure, VisHeader, lists the elements written to the header record. This is 1 header record followed by multiple VisPixel data records.

The VisHeader Declaration

```
 TYPE VisHeader
   INTEGER                                                          :: NLinesSkipped
   INTEGER                                                          :: NPixelPerLine
   INTEGER                                                          :: NLines
   INTEGER                                                          :: NOutputs
   INTEGER (int1)                                                   :: BytesPerPixel
   CHARACTER (LEN=MaxChar), DIMENSION(MaxValues)                    :: OutputDesc
   INTEGER (int1),        DIMENSION(MaxValues)                      :: OutType
   REAL,              DIMENSION(MaxValues, MaxLimits)               :: OutLimits
   INTEGER (int1),        DIMENSION(MaxValues)                      :: NLevels
   CHARACTER (LEN=MaxChar), DIMENSION(MaxValues, MaxLevels)         :: OutputLDesc
 END TYPE VisHeader
```

The VisPixel Declaration

```
 TYPE VisPixel
   INTEGER (int1)          :: AVHRRLon_00
   INTEGER (int1)          :: AVHRRLon_01
   INTEGER (int1)          :: AVHRRLat_00      ! this is co-lat
   INTEGER (int1)          :: AVHRRLat_01      ! this is co-lat
   INTEGER (int1)          :: Chan1Ref
   INTEGER (int1)          :: Chan3BTM
   INTEGER (int1)          :: Chan4BTM
   INTEGER (int1)          :: DayFlag
   INTEGER (int1)          :: SercaaMask
   INTEGER (int1)          :: WelchCloudMask
   INTEGER (int1)          :: WelchDesertMask
   INTEGER (int1)          :: WelchSnowMask
   INTEGER (int1)          :: WelchSmokeMask
   INTEGER (int1)          :: FinalMask
   INTEGER (int1)          :: ClassifyLayer
   INTEGER (int1)          :: StoweAOTProperty
   INTEGER (int1)          :: VintCloudFraction
   INTEGER (int1)          :: VintEffOpticalDepth
   INTEGER (int1)          :: VintEmissivity
   INTEGER (int1)          :: VintWaterPath
   INTEGER (int1)          :: VintTopPressure
   INTEGER (int1)          :: VintEffPressure
   INTEGER (int1)          :: VintEffTemp
   INTEGER (int1)          :: VintEffHeight
   INTEGER (int1)          :: VintBottomPressure
   INTEGER (int1)          :: VintParticalRadius
   INTEGER (int1)          :: VintParticalPhase
   INTEGER (int1)          :: PlatnickOptThick
   INTEGER (int1)          :: PlatnickEffRadii
   INTEGER (int1)          :: AirMass
   INTEGER (int1)          :: MOASurfTemp
   INTEGER (int1)          :: CondensedType
 END TYPE VisPixel
```

The data structure, CloudValPixel, lists the data elements written to the file CloudVal.

CloudValPixel Declaration

```
integer, parameter                  :: NumLayers      = 2
INTEGER CloudValUnit
CHARACTER (len =200)                :: CloudValFile

 TYPE CloudValPixel
   REAL, DIMENSION ( NumChan )      :: rad
   REAL, DIMENSION ( NumChanVIS )   :: reflec
   REAL, DIMENSION ( NumChanIR )    :: BTemp
   REAL                             :: time
   REAL                             :: lat
   REAL                             :: lon
   REAL                             :: SolZen
   REAL                             :: azimuth
   REAL                             :: ViewZen
   REAL                             :: elevat
   REAL                             :: CSalbedo
   REAL                             :: CSbtemp
   INTEGER                          :: condensed
   INTEGER                          :: ecosys
   INTEGER                          :: SercaaMaskRes
   INTEGER                          :: WelchMaskRes
   INTEGER                          :: FinalMaskRes
   INTEGER                          :: ClassifyLayerRes
   INTEGER                          :: SpaCohLayerRes
   INTEGER                          :: FinalLayerRes
 END TYPE CloudValPixel
```

Appendix C

Data and "Constants"

Appendix C - Data and "Constants"

The protopixel is the structure to hold the imager values in a consistent manner regardless of which imager is input to the system. For Release 1, the protopixel reflects the number of channels from the AVHRR test data set and the VIRS data set. During Release 2, the number of MODIS channels required by the Science Team will be reflected in the protopixel and the master pixel.

The "ProtoPixel" declaration

```
INTEGER, PARAMETER :: NumChan    = 5
INTEGER, PARAMETER :: NumChanVIS = 3
INTEGER, PARAMETER :: NumChanIR  = 3

TYPE ProtoPixelType
   REAL,    DIMENSION ( NumChan )      :: rad
   REAL,    DIMENSION ( NumChanVIS ) :: ref
   REAL,    DIMENSION ( NumChanIR )   :: btm
   INTEGER, DIMENSION ( NumChan )    :: counts
   REAL                               :: lat
   REAL                               :: lon
   REAL                               :: time
   INTEGER                            :: sat
   INTEGER                            :: year
   INTEGER                            :: day
END TYPE ProtoPixelType
```

The "master" pixel declaration, CloudPixel

```
TYPE CloudPixel
  INTEGER                          :: Flag_Count
  INTEGER                          :: Flag_Rad
  INTEGER                          :: Flag_Ref
  INTEGER                          :: Flag_Btm
  INTEGER                          :: Flag_dummy      ! to keep record size the same as Prototype
  INTEGER, DIMENSION(NumChan)      :: COUNT
  REAL,    DIMENSION(NumChan)      :: Radiance
  REAL,    DIMENSION(NUmChanVis)   :: Reflect
  REAL,    DIMENSION(NumChanIR)    :: BrightTemp
  REAL                             :: SolarZen
  REAL                             :: ViewZen
  REAL                             :: Azimuth
  REAL                             :: Elevation
  REAL                             :: time
  REAL                             :: Latitude
  REAL                             :: Longitude
  REAL                             :: CSalbedo        ! Clear sky albedo
  REAL                             :: CSreflect       ! Clear sky reflectance
  REAL                             :: CSbrightTemp    ! Clear sky bright Temperature
```

```
REAL                        :: SunGlint          ! SunGlint Test
  INTEGER                   :: sat
  INTEGER                   :: Year              ! data year
  INTEGER                   :: day               ! date day of the year
  INTEGER                   :: Condensed         ! Condensed map
  INTEGER                   :: EcoSys            ! Eco system
  INTEGER                   :: WaterCont         ! Water contents
  INTEGER                   :: ErbSID            ! ERBS scene id
  INTEGER                   :: SnowMap
  INTEGER                   :: IceMap
  INTEGER                   :: DayFlag           ! Day/night flag
  INTEGER                   :: PolFlag           ! Polar flag
  INTEGER                   :: MOAIndex
  INTEGER                   :: SercaaMaskRes     ! SERCAA test result
  INTEGER, DIMENSION(WelchValues) :: WelchMaskRes   ! Welch test result
  INTEGER                   :: FinalMaskRes       ! cloudy = 1, clear = 0
  INTEGER                   :: ClassifyLayerRes  ! Baum classification test result
  INTEGER                   :: SpaCohLayerRes    ! Coakley test result
  INTEGER                   :: FinalLayer         ! Final cloud layer result
  REAL                      :: StoweAOTRes       ! Stowe
  REAL, DIMENSION(PlatnickValues) :: PlatnickRes       ! Platnick test result
  REAL, DIMENSION(VintNumLayers,VintValues)        :: VintPropertyRes
  INTEGER (INT1)            :: Rutan
END TYPE CloudPixel
```